VYSOKÁ ŠKOLA BÁŇSKÁ — TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY, KATEDRA INFORMATIKY

Knihovna filtrovacího jádra firewallu implementující detekci napadení
Firewall Library Implementation for Intrusion Detection

DIPLOMOVÁ PRÁCE

Miloslav Serba

jaro 2010

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Ing. Martin Milata

Poděkování

Na tomto místě bych rád poděkoval několika lidem, kteří výrazným způsobem podpořili tuto práci. Děkuji

- Ing. Karlu Oblukovi, Ph.D., za umožnění realizace práce v rámci AVG IS,
- Mgr. Jiřímu Brackovi za technické rady a tipy z oblasti Windows NT,
- Ing. Zdeňku Poláchovi za technické rady a tipy z oblasti počítačových sítí,
- Ing. Martinu Milatovi za vedení tohoto projektu.

Shrnutí

Firewall, jako základní programové vybavení pro posílení bezpečnosti počítače, aktivně brání vzniku bezpečnostního incidentu nebo pasivně hlásí jeho přítomnost. Cílem této práce je popsat a vytvořit subsystém pro operační systém Windows NT v rámci komerčního balíku AVG Internet Security, který bude informovat o některých rizicích na počítačové síti, případně jim předcházet. Práce se zaměřuje na protokoly ARP a TCP, popisuje obecně i konkrétně problémy těchto protokolů a navrhuje případná řešení.

A firewall is considered a basic security software responsible for an active protection from a security breach and a passive report of an incident. The goal of the thesis is to create and describe a subsystem for the Windows NT operating system family inside an existing commercial application AVG Internet Security. The subsystem will inform a user about some risks in computer networks and it will avoid them if possible. The thesis is focused on ARP and TCP, it describes their problems generally and specifically, and proposes possible solutions.

Klíčová slova

firewall, bezpečnost, security, filtrace, filtering, TCP/IP, ARP, útok, attack, napadení, intrusion, ochrana, protection, počítačová síť, computer network, Windows kernel driver, NDIS, TDI, IDS, IPS, Windows NT

Obsah

Kapitola 1

Úvod

Před vývojem samotných ochranných opatření proti nebezpečím síťového původu na Internetu je nutné pochopit, jak operační systém s provozem zachází, jak je daný provoz definován a kde se nachází jeho bezpečnostní nedostatky.

Proto je potřeba znát hlavní rysy operačního systému, prostředky, které nám k tomuto účelu nabízí, a způsob zpracovávání síťových požadavků v systému. Tímto popisem prostředí se zabývá kapitola 2.

Popis jednotlivých protokolů, jejich slabá místa, možnosti filtrování, detekce a ochrana proti těmto hrozbám bude hlavním tématem další kapitoly, každá podkapitola bude zaměřena na jeden druh komunikace. V kapitole 6 si přiblížíme řešení v jiných nástrojích podobného zaměření.

V kapitole 4 se zaměříme na implementaci, popíšeme, jak systém ochrany funguje. V následující kapitole si ukážeme další možné kroky pro zdokonalení ochrany.

Kapitola 2

Architektura Microsoft Windows NT z pohledu firewallu

2.1 Kernel space

Pojem `kernel space` (prostředí jádra, kernel) označuje část OS, ve které běží veškeré ovladače. Běh v kernelu má několik specifických pravidel. Některé z nich je vhodné na tomto místě zmínit, jejich důsledky totiž zasahují do návrhu naší filtrační knihovny.

Kernel mód nerozlišuje mezi běžícími vlákny vlastníky, tedy každý běžící ovladač má možnost dostat se k jakémukoliv systémovému prostředku. Nepříjemným důsledkem této jinak vynikající možnosti je fakt, že nikdo nás neomezuje v přístupu do paměti, čili je velice jednoduché vlivem chyby zapsat na místo patřící jinému vláknu, v nejhorším případě do kritické části systému, což v drtivé většině případů vede k zastavení systému. To se projevuje známou „modrou obrazovkou smrti“ (BSOD, `Blue Screen Of Death`).

2.1.1 Přepínání kontextů, virtuální paměť, stránkování

Tři výše jmenované oblasti silně ovlivňují vývoj ovladačů pro kernel. Ovladače jsou nahrané ve fyzické paměti, tady závisí pouze na operačním systému, která aplikace je nahrána v které části paměti. Za tímto účelem je provoz v kernelu (pro zjednodušení nepočítáme-li obsluhu hardwaru) rozdělen do dvou režimů, které odpovídají úrovním přerušení: první se označuje pasivní a druhá (nad ní) výkonná (angl. `passive level` a `dispatch level`). V pasivním režimu máme možnost dosáhnout na virtuální paměť běžících procesů, a to proto, že systém (jako kernel ovladač) zajišťuje stránkování právě v režimu výkonném (běží na vyšší úrovni přerušení). Stejně tak v tomto režimu systém zajišťuje přepínání procesů, synchronizaci prostředků a obsluhu přerušení. Omezení, která tímto klade na druhý režim, jsou následující:

- stránky paměti mohou být na disku, je tedy nutné je před použitím načíst do fyzické paměti a zamknout (běžíme na stejné úrovni přerušení jako část systému, která se o stránkování stará, tzn. přerušili jsme její činnost),
- zamknutím sdíleného prostředku dosáhneme bezproblémové synchronizace, ovšem zároveň znemožníme přepínání procesů a stránkování, proto (až na výjimky) nelze čekat na synchronizační prostředek,
- fyzická paměť je vzácný zdroj, kterým je potřeba šetřit.

Odměnou za dodržení těchto podmínek je mj. téměř nepřerušitelný běh vlákna neboli jinak řečeno plný výpočetní výkon jednoho procesoru.

2.1.2 Alokace paměti

Alokace paměti je rozdělena (právě podle režimů) do dvou základních prostorů. Těmi jsou `paged` a `nonpaged` (česky stránkovaný a nestránkovaný). Již jsme zmínili, že stránkování probíhá v dispatch režimu, tedy stránkovaná paměť (teoreticky neomezená) není standardně přístupná (resp. její existence ve fyzické paměti není zaručena) v režimu výkonném. Naopak paměť nestránkovaná je vždy uložena ve fyzické paměti, je vždy celá přístupná všem ovladačům a je značně omezená (dle instalované paměti v počítači). Z důvodu rozdělení paměti na dva tábory se i funkce jádra musí logicky rozdělit dle možností užití. Ty první jsou schopny běžet pouze v rámci fyzické paměti, nepřistupují do stránkované paměti, a proto mohou běžet na úrovni dispatch. Ty ostatní pro svoje vykonání vyžadují více paměti a je nutné je spouštět nad pamětí stránkovanou, tzn. tyto funkce nesmí za žádných okolností běžet na úrovni dispatch. Porušení tohoto pravidla by vedlo k nutnosti načíst stránku, ovšem v režimu dispatch by nedošlo k obsloužení této výjimky a systém by oznámil `BSOD`literal> chybu.

2.1.3 Přechody mezi úrovněmi

Mezi jednotlivými úrovněmi passive a dispatch lze přecházet. Protože dispatch úroveň zamezuje přepínání vláken, máme vstupem do této úrovně zaručeno, že naše vlákno nebude po tuto dobu přerušeno (toho využíváme především při zamykání sdílených prostředků, při zpracovávání přerušení apod.). Z tohoto důvodu je nutné čas strávený ve výkonném módu využít výhradně pro synchronizačně náročné úlohy a omezit dobu běhu na nezbytně nutnou. Je jasné, že v tomto režimu nelze čekat na jiné synchronizační události, pakliže jejich vyvolání není způsobeno na vyšší úrovni, než je úroveň výkonná. Počet právě běžících vláken je dán počtem jader procesoru (resp. počtem procesorů). Provádění v dispatch režimu tudíž neznamená, že nemůže běžet žádné jiné vlákno. Pouze v případě, že obě vlákna na dispatch úrovni se mají synchronizovat pomocí stejného prostředku (nejedná-li se o pouze o nastavení události), pak jedno z vláken bude aktivně čekat na dokončení operace druhého vlákna, které tento prostředek získalo. V rámci dobrého návrhu je vhodné právě těmto situacím předcházet, protože tím blokujeme výpočetní výkon a zároveň neumožňujeme neaktivnímu jádru přejít do úspornějšího režimu.

Nutnost pracovat s těmito úrovněmi vzniká z podstaty fungování operačního systému. Systém využívá výkonnou úroveň z důvodů vyššího výkonu. Pokud zrovna s výkonnou úrovní je zavolán náš ovladač, je nutné dodržet pravidla na této úrovni platná. Protože se tím zmenšuje prostor funkcí, které je možno zavolat, systém nabízí možnost aktuální běh v naší části ovladače pozastavit a s využitím pomocného vlákna posléze kód vykonat na úrovni pasivní.

2.2 Dvojvrstvý model síťové filtrace

Operační systém poskytující plnou síťovou podporu je navržen tak, že na každé úrovni v systému (odpovídající jisté vrstvě ISO OSI [10] modelu) je možné sledovat případně interagovat s procházejícími daty specifickými pro danou vrstvu. Tím je umožněno např. vytvoření vlastního ovladače síťové karty na spodní vrstvě nebo přijímat datový tok v aplikaci na horní vrstvě. Tento model nám nabízí několik míst pro kontrolu a ovlivňování síťového provozu. Našim cílem je zvolit takové vrstvy pro firewall, abychom měli nejen dostatek informací o datech, která hodláme filtrovat, ale abychom také data zachytili ve formě, která umožní jejich prověření. Je jasné, že spodní vrstvy (tj. blíže k síťové kartě) jsou blíže přenášeným datům, kdežto horní vrstvy (blíže k aplikaci) budou nabízet více `metadat` o daném provozu.

Mezi vrstvy (pozice) vhodné či používané pro zařazení firewallu (ať již části nebo celého filtrovacího procesu) patří tyto:

- LSP,
- ovladač mimo síťové vrstvy,
- TDI,
- ovladač sítového filtru,
- NDIS,
- WFP.

2.2.1 LSP fronta

LSP je anglická zkratka pro `Layered Service Provider` [12], česky lze označit `vrstvený poskytovatel služeb` (doslovného překladu se neužívá, používá se LSP fronta či LSP seznam). Jedná se o rozšíření síťové podpory Windows umožňující zapojení knihoven třetích stran do systémové části zajišťující provoz nad protokolem IP [7] resp. TCP/IP [22] [23] (dále jen `TCP/IP stack`). Výhodou tohoto použití je jeho relativně snadná implementace (co do rozsahu) a přístup k provozu IP. Obrovskou nevýhodou je zapojení knihoven do LSP seznamu. Chyba jedné knihovny ovlivňuje ostatní a navíc poškozením tohoto seznamu se uživatel dostává do problémů s připojením k Internetu obecně.

Bezpečnostní aplikace využívající v minulosti LSP seznam od tohoto rozhraní upustily, většinou při ukončení podpory systémů Windows rodiny 9X. Důvodem byly nejenom jmenované potíže, ale i použití vhodnějších částí v rodině NT.

2.2.2 Ovladač mimo síťové vrstvy

Tímto máme na mysli obecný ovladač, který využívá přímého přístupu k síťovým ovladačům a jejich funkcím, avšak sám není přímo zapojen do síťových operací.

Výhodou tohoto zapojení je větší nezávislost v implementaci a možná dostupnost většího počtu informací o filtrovaném obsahu. Na straně záporů vede nutnost údržby teoreticky odlišného kódu pro všechny podporované mutace operačního systému, dále přístup přes interní nebo nedokumentované funkce (mohou se bez předchozího upozornění měnit), větší náročnost na uspořádání do uceleného souboru (větší implementační nároky).

Právě odloučení od síťového modelu činí tento způsob zapojení samostatně nepoužitelný. Avšak v souvislosti s jiným přístupem můžeme hovořit o schůdném řešení.

2.2.3 TDI ovladač

Z anglického označení `Transport Driver Interface` [25], česky volně přeloženo jako „rozhraní přenosového ovladače“ (v češtině se užívá označení TDI). Označení TDI je často používáno pro popis rozhraní (`TDI událost`) i pro implementace v podobě ovladačů (`TDI ovladač`). Rozhraní TDI je vrstva mezi operačním systémem (tj. nad TCP/IP stackem) a aplikačním prostorem.

Celé TDI rozhraní je postavené na událostech, které dostávají všechny TDI ovladače v systému a to v pořadí v jakém byly připojeny k TCP/IP stacku (tj. k TCP/IP ovladači v systému, přesněji k jeho zařízením pojmenovaných podle síťových protokolů). Toto pořadí lze ovlivnit, ale pokud TDI ovladač zpracovává události tak, že brání korektní funkci druhého ovladače, opět mluvíme o nedostatku jako u LSP. Události (doplněné o nezbytné parametry) lze přirovnat k oznamování kroků při vytváření spojení, odesílání a příjmu dat, ukončování spojení. Z tohoto přístupu plyne nevýhoda v podobě nedostupnosti síťových dat, pomocí TDI jsme schopni zpracovávat pouze aplikační data (tj. máme data z aplikace, nikoliv data později dodaná systémem a např. odeslaná z počítače). TDI ovladačů může existovat v systému několik, proto se zařazují tak, že zastoupí původní událost vlastním voláním a na závěr zpracování ji postoupí dalšímu ovladači. Z tohoto důvodu nelze již jednou zavedený ovladač z této fronty událostí vyřadit (což nemusí vadit, protože „vypnutý“ ovladač bude události bez dalšího zpracování pouze přeposílat). TDI rozhraní nenabízí žádné vlastní specifické prostředky pro filtraci provozu ve smyslu pravidel nebo rozhodovacích procesů, tuto podporu je nutné zcela implementovat ve vlastní režii.

Detailní informace a přesně popsané rozhraní dělá z TDI vhodný prostředek pro vytvoření filtrovací základny. Aktuálně TDI používá několik bezpečnostních programů, které bychom mohli nazvat „aplikační filtry“. Je až s podivem, kolik aktuálně nabízených produktů pro zabezpečení počítače se spokojí pouze s aplikačním filtrem a surová data odcházející/přicházející do systému ignoruje. V případě příchozího provozu jsou pakety zachyceny těsně před aplikací, jsou tedy již zpracovány systémem, což může být bezpečnostním rizikem. Stejně tak odchozí pakety na úrovni systému (odesílané přímo pomocí `tcpip`) nemusí být touto vrstvou zachyceny, což propustí např. síťový provoz rootkitu. Je nanejvýš

vhodné doplnit filtrování na TDI vrstvě ještě o spodnější vrstvu (blíže k síťové kartě a surovým datům).

S příchodem přepracovaného TCP/IP stacku na operačním systému Windows Vista se zásadně mění TDI vrstva. Vista již přímo nepodporuje toto rozhraní a jeho zpracování se provádí pomocí emulace se všemi negativy, které emulace obecně přináší. Systém Windows 7 je posledním systémem, který nějakým způsobem TDI podporuje [26]. Nástupcem TDI a řešením pro filtrování síťového provozu na systémech rodiny NT má být technologie WFP popsaná dále.

2.2.4 Ovladač sítového filtru

Síťové rozhraní v operačním systému Windows NT umožňuje zapojit do zpracování provozu ovladače, které poskytují filtrovací (rozhodovací) funkci, v angličtině ozn. `filter hook` ovladače. Ty mají přístup k datům přímo z TCP/IP a zpracovávají tedy pakety. Výhodou tohoto řešení je jednoduchá implementace. Bohužel, systém zapojení do systému opět naráží na základní nedostatek spojený s pořadím filtrů a neschopností operačního systému sdružovat výsledné verdikty skrz zapojené ovladače. Řešení je velice podobné LSP ovladačům.

2.2.5 NDIS

NDIS (`Network Driver Interface Specification` [14]), volně nazváno „specifikace rozhraní síťového ovladače“ je předpis z dílny společnosti Microsoft určující komunikační kanály mezi logickými objekty odpovídající síťové kartě a operačnímu systému. Jedna či více síťových karet komunikují s jednou či více síťovými kartami nebo s jedním či více síťovými protokoly nad nimi. Pro spodní vrstvu objektů (síťové karty) se používá označení `miniport` („brána, dveře“) pro vrchní vrstvu `protocol` (protokol). Mezi nimi lze umístit další logické objekty, které se nazývají `intermediate` (prostředník). Každá úroveň NDIS ovladače odpovídá jinému typu použití. Miniport je zařízení schopné přístupu k zařízení, protokol má přístup k paketům a intermediate umí komunikovat s oběma rozhraními nad i pod, což je zejména výhodné pro bezpečnostní software.

NDIS je často zaměňován za konkrétní implementaci některého z výše uvedených objektů. Ve skutečnosti operační systém poskytuje NDIS rozhraní nad svými vrstvami a funkcemi. Takže v rámci implementace objektu pomocí NDISu máme k dispozici kompletní sadu nástrojů pro základní operace a nemusíme duplikovat funkcionalitu systému i ve svém ovladači.

Na rozdíl od vrstvy TDI se NDIS stále vyvíjí. V operačním systému Windows 2000 měl verzi 5.0, ve Windows 7 je to již 6.1. Právě přechod od verze 5.1 ve Windows XP k verzi 6.0 ve Windows Vista znamenal významné ulehčení pro bezpečnostní software. Ve verzi 6.0 vzniká nový logický objekt. Je jím `filter` (filtr). Je umístěn obdobně jako prostředník, ale má velmi zjednodušené rozhraní, takže je snazší implementovat ovladač monitorující provoz mezi miniportem a protokolem.

2.2.6 WFP

Novinkou operačního systému Vista je technologie WFP (`Windows Filtering Platform` [27]). Právě nutnost používat dvoje rozhraní (TDI pro spojení a NDIS pro pakety) vedla k vytvoření systému, který má celý proces filtrování provozu zjednodušit. Přínosem je možnost filtrace spojení na různých úrovních, což má právě nahradit přístup přes TDI a NDIS. Navíc je možné zapojit několik filtrovacích programů.

Technologie WFP je plně zapojena do přepracovaného TCP/IP a je možné ji rozšiřovat jak v uživatelském prostoru, tak i v jádře. Stejně jako u NDISu je WFP doplněna řadou systémových prostředků, které usnadňují filtrování provozu. WFP ale není firewall, je to pouze prostředek, jak rychle nějaký firewall vyvinout. Právě Windows Firewall na systémech Vista a Server 2008 je plně implementován jako klient WFP.

Jediným bezpečnostním rizikem je samotný přístup WFP. Jestliže ponese jednu chybu, touto chybou budou trpět všechny produkty WFP využívající. Na druhou stranu tlak výrobců bezpečnostního softwaru bude v těchto případech silnější a oprava se objeví dříve, než za aktuálního stavu, kdy na chyby se přichází spíše izolovaně.

2.2.7 Celkový náhled na jednotlivé síťové vrstvy v systému

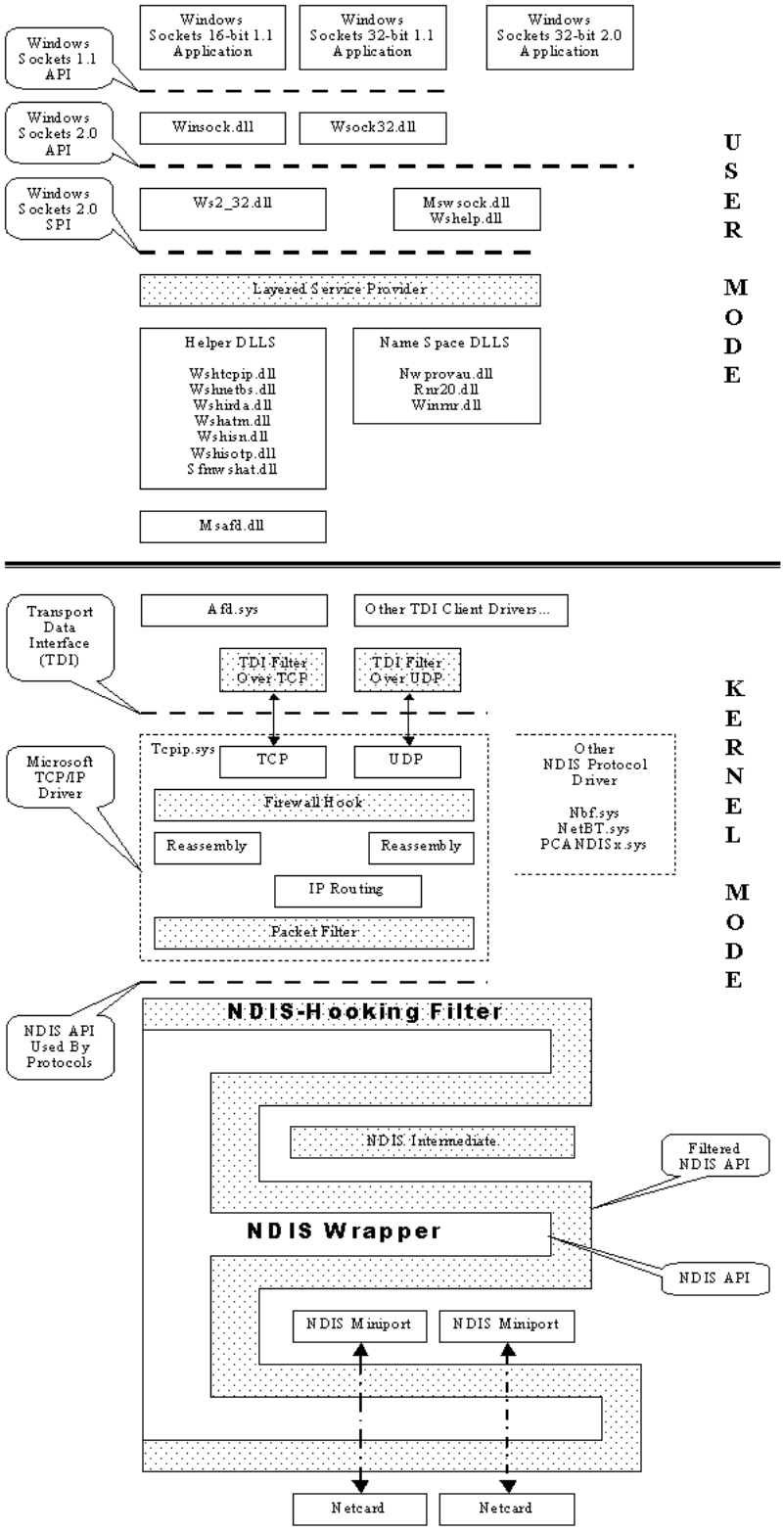
Následující obrázek[14] 2.1 ukazuje umístění síťových vrstev na systému Windows XP (chybí zde vrstva WFP a NDIS má verzi 5.1).

Rozdíl mezi NDIS 6.0 [13] filtrem (obrázek 2.2) a prostředníkem (obrázek 2.3). Filtr je pro horní i dolní vrstvu transparentní.

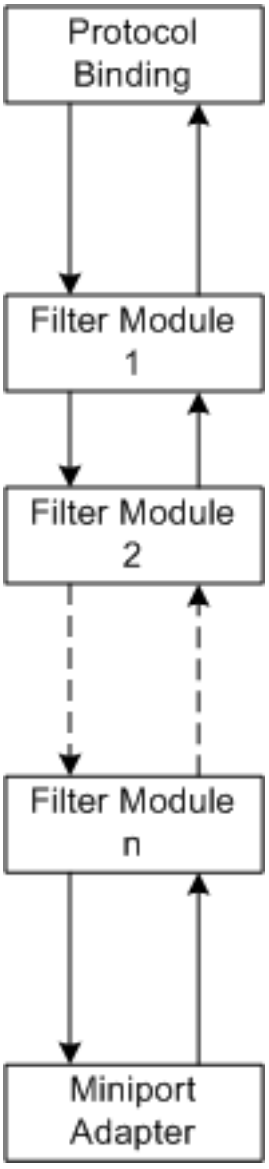
Přehled WFP architektury[27] (obrázek 2.4).

2.2.8 Výběr filtrovací technologie

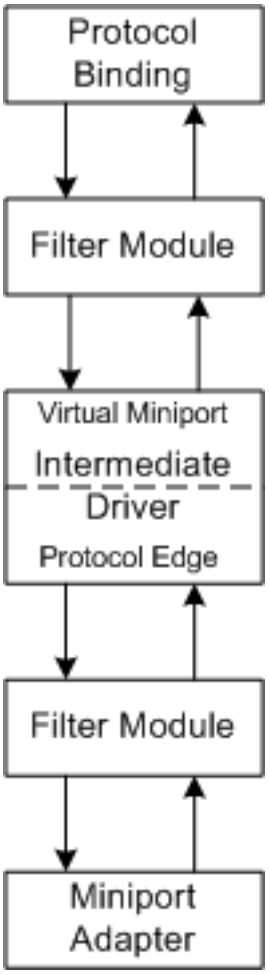
Z dostupných filtrovacích technologií se jeví jako nejlepší řešení použít technologii WFP. Ačkoliv to u ní nebylo zmíněno (protože se jedná o nefunkční požadavek), jejím nedostatkem je nedostupnost na systémech předcházejících OS Vista. Naším cílem je dosáhnout filtrační schopnosti v celé rodině NT (počínaje Windows 2000), je proto nutné zvolit druhé nejlepší řešení: kombinaci TDI a NDIS ovladače. Spolu s tímto řešením přichází omezení pro systémy Vista a vyšší, ale jak si později ukážeme, lze ho úspěšně obejít (zastaralá TDI část bude nahrazena přístupem přes WFP). Filtrační proces tak bude rozdělen na dvě části: paketový (zajištěný pomocí NDISu) a aplikační (zajištěn skrze TDI).



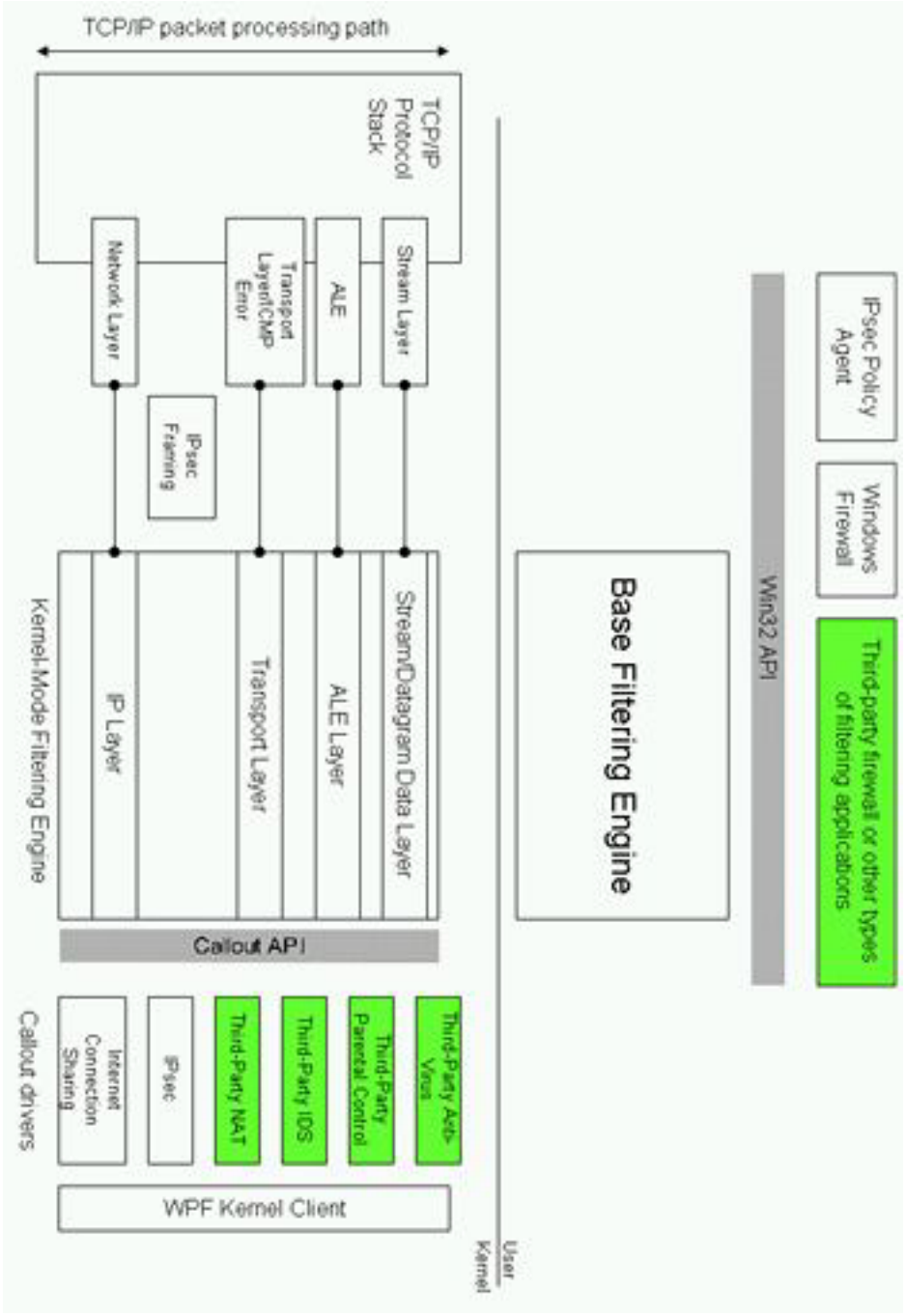
Obrázek 2.1: Síťové vrstvy v systému (aplikační a systémová část)



Obrázek 2.2: NDIS filter (filtr)



Obrázek 2.3: NDIS intermediate (prostředník)



Obrázek 2.4: WFP technologie

Kapitola 3

Síťové protokoly

3.1 ARP

Zkratka pochází z anglického jména `Address Resolution Protocol` [1] (protokol překladu adres). ARP je Ethernetovým[6] protokolem v síťové vrstvě[1] modelu ISO OSI. Spolu s protokolem IP[7] tvoří základní prvek pro komunikaci mezi počítači.

3.1.1 Použití protokolu

ARP používá operační systém k učení fyzické (hardwarové, linkové) adresy příjemce z jeho adresy logické (síťové). V případě protokolu IP se ARP používá pro zjištění fyzické (hardwarové, MAC[6]) adresy z adresy IP. Doplňkem ARPu je protokol RARP[15] (zpětný ARP), který se používá pro zjištění IP adresy z hardwarové adresy (čehož se využívalo např. při spouštění počítače ze sítě). ARP protokol je používán lokálně, tzn. (ve výchozím nastavení) neprochází směrovači (routery), protože mezi jednotlivými segmenty sítě se ke směrování používá právě síťová vrstva ISO OSI.

ARP provoz se může používat i k účelům, pro které nebyl původně navržen. Jedná se např. o distribuci zátěže mezi několik strojů, kdy pro ARP dotazy z různých IP segmentů odpovídají jiné stroje s jinou MAC adresou (ARP paket s odpovědí je vygenerován pro totožnou IP adresu, ale různými stroji). V reálu tak dochází k pomyslnému ARP útoku, kdy se několik počítačů vydává za tentýž stroj s jednou IP adresou. Toto řešení není optimální, pro rozdělení zátěže provozu (tzv. `load balancing`) se aktuálně používají standardnější metody jako přepínače (switche) nové generace (schopné uchovávat stavové informace) na straně hardware či DNS[5] na straně software. Použití vedlejších efektů ARP pro úlohy neodpovídající původnímu záměru omezuje nasazení striktního filtrování, protože pro každé nestandardní chování je potřeba zavést do filtrovacího procesu výjimku. Výsledný systém se tak stává sice více univerzálním, ale méně bezpečným. Navíc není v lidských silách sledovat veškerá postranní řešení a zavádět vynucené „opravy“ do systému.

Komunikace pomocí ARP paketů probíhá mezi dvěma zařízeními, případně na všechna zařízení na podsíti (tzv. `broadcast`, všesměrové vysílání). Z tohoto pohledu rozlišujeme mezi dvěma typy zpráv: dotaz a odpověď. Pokud chce počítač s IP adresou `S=2.2.2.2`

1. Pokud by někdo namítal, že samotný ARP není závislý na IP adresách a ty jsou pro něj pouze „výplní paketu“, tedy ARP musí být na linkové (spojovací) vrstvě, jako argument pro síťovou vrstvu uveďme teoretickou nezávislost Ethernetu na ARPu a dále fakt, že ARP je sestavován za Ethernet hlavičku.

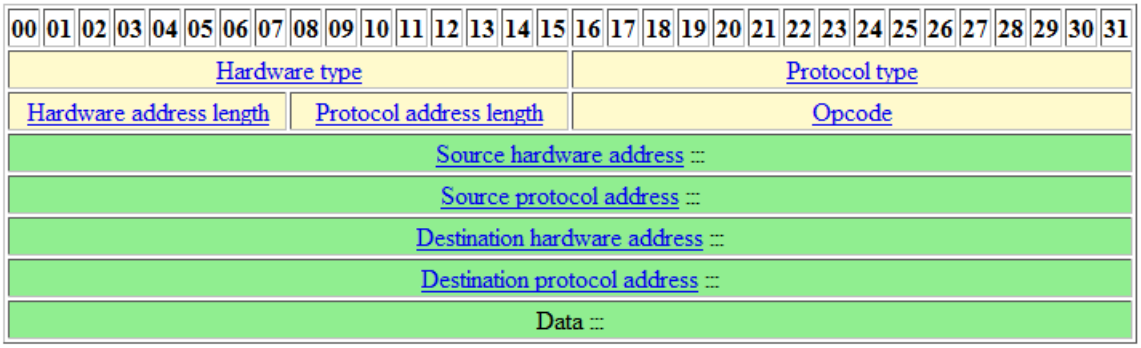
poslat data příjemci s IP adresou `R=1.1.1.1`, pak odešle ARP dotaz všem klientům, kterých se ptá „Kdo má adresu R, odpovězte S.“. Jestliže dotaz dorazí stroji s IP adresou R a MAC adresou M, vytvoří odpověď „R je dostupná na adrese M“, kterou však odešle přímo tazateli S. Tímto způsobem první počítač S získá cílovou fyzickou adresu příjemce R.

Povinnost počítače odpovědět na ARP dotaz s odpovídající fyzickou adresou se využívá také při přidělování IP adresy. Po získání adresy stroj odešle ARP (broadcast) dotaz na svoji fyzickou adresu. Pokud obdrží alespoň jednu odpověď, na síti existuje konflikt IP adres. Takto použité ARP dotazy se označují `Gratuitous ARP`. I přes to, že se jedná o jednoduchý a rychlý způsob ověření duplicitní IP adresy, postup ještě není standardizován[**?**].

Z důvodů ušetření přenášených dat na síti se operační systémy vybavují vyrovnávací pamětí (`ARP cache`) tvořené z dvojice IP a MAC adresa. (Broadcast odpověď se naopak může používat pro šíření této cache a snížení lokálního provozu.) Právě použití cache (jak bude dále popsáno) je kompromisem mezi výkonem a bezpečností.

3.1.2 Obsah paketu

Graficky znázorněný paket ARP[17] nalezneme na obrázku č. 3.1. Z pohledu firewallu na operačním systému Windows NT nás zajímají pouze ty pakety ARP, jejichž `Hardware type` má hodnotu `1` (Ethernet) a `Protocol type` odpovídá `0x800` (IP). `Opcode` musí obsahovat hodnoty `1` (dotaz) či `2` (odpověď). ARP pakety jiných konfigurací nejsou pro filtrování ARP spolu s IP provozem na síti typu Ethernet důležité a mohou být ignorovány. Vyobrazený paket neobsahuje data z protokolů nižších vrstev (Ethernet).



Obrázek 3.1: Paket ARP

3.1.3 (Ne)bezpečnost protokolu

ARP neobsahuje žádné bezpečnostní mechanizmy, paket nenese věrohodnou nepopiratelnou informaci o odesílateli. Cílem útočníka je vystupovat jako `man-in-the-middle` pro oběť a první routovací zařízení. Při úspěšném útoku napadený počítač směruje provoz na útočníka místo na směrovač a směrovač naopak posílá data útočníkovi místo napadenému počítači. Útočník napadá stanice dvěma způsoby:

- Odesláním podvrženého ARP dotazu.
- Odesláním podvržené ARP odpovědi.

V obou případech se oběti (resp. routeru) odešle upravený ARP paket, kde jako IP adresa odesílatele figuruje IP adresa směrovače (resp. oběti), ale MAC adresa odesílatele (původně směrovače resp. napadeného) je nahrazena MAC adresou útočícího stroje. V nechráněném prostředí dojde k tomu, že veškerý provoz původně přenášený mezi počítačem a směrovačem je veden přes prostředníka. Ten tak má možnost sledovat veškerý provoz mezi těmito dvěma body. Vytvoření podvržených paketů není nikterak složité, napadené stroje mezi sebou nesdílí dopředu ani v průběhu ARP komunikace žádné zabezpečující prvky. Pro zahájení útoku stačí znát cílové IP adresy a odeslat dva (v případě podvržené odpovědi čtyři) ARP pakety. Protože do celého procesu napadení vstupují i korektní odpovědi napadených, je potřeba eliminovat problém načasování podvržených odpovědí jejich opakovaným odesíláním, čímž se docílí situace, že správně použité hardwarové adresy se ihned přepisují na infikované.

Do procesu ARP útočení vstupuje ještě vyrovnávací paměť operačního systému na ARP záznamy (jak již bylo naznačeno). V tomto případě je ovšem její použití spíše kontraproduktivní, protože snižuje četnost ARP dotazů/odpovědí na síti, takže útočníkovi stačí odhalit časový interval vypršení platnosti záznamu v ARP cache a není nutné neustále odesílat podvržené pakety a je možné odesílací interval mezi jednotlivými podvrženými pakety prodloužit.

Napadení mohou znesnadňovat přepínače (switche) umístěné mezi obětí a směrovačem. Přepínače mohou omezovat provoz dle hardwarových adres, což by vždy odřízlo útočníka od přístupu k napadenému v případě odeslaného korektního ARP paketu ze směrovače/klienta. Naneštěstí tento systém je závislý na uchovávání informací o dosavadním přenosu a v případě zaplnění paměti se switch[2] začíná chovat jako pouhý „přeposílač“ a útočník má volnou cestu k oběti.

V anglických textech se pro tento typ útoku používá pojmenování `ARP spoofing` (podvržení ARPu). Synonymem je též označení `ARP cache poisoning` (infikování ARP vyrovnávací paměti). ARP útok je většinou první prostředkem, který útočník vyzkouší, protože se jedná o jednoduchý a v případě úspěchu velmi účinný nástroj.

3.1.4 Ochrana před útoky

Detekce podvrženého ARP paketu není až tak jednoduchá, právě kvůli strohosti samotného protokolu. Přenášeny jsou pouze nejdůležitější informace, takže filtrování na základě samotného obsahu paketu není možné. Ochranou proti podvrženým ARP paketům je sledování dosavadního provozu a uchovávání informací o tom, kdo používá jakou fyzickou adresu a IP adresu. Nevyžádaná data jsou považována za nedůvěryhodná a nevedou k uchová-

2. V tomto případě máme na mysli jednoduché zařízení poskytující pouze funkcionalitu switche. Moderní, dražší přístroje typu switch již disponují prostředky pro zamezení popisovaného typu útoku.

vání informací. V případě konfliktu se spouští analýza shromážděných záznamů a hledá se podezřelý prvek.

Při napadení podvrženým ARP dotazem je postup následující:

- Příchozí dotaz je zpracován systémem.
- Ze systémové cache je odebrána informace o odesílateli.
- Systém odešle ARP odpověď.
- Systém odešle ARP dotaz odesílateli a poznamená si, že čeká na ARP odpověď.
- Systém přijme ARP odpověď, a protože existuje záznam o čekání na odpověď, ta je povolena.

Za normálního provozu je MAC adresa odesílatele uložena v ARP cache. V případě chráněného prostředí tak dochází k nárůstu ARP provozu, protože v případě spojení s odesílatelem je potřeba se na jeho fyzickou adresu opět dotázat (což je provedeno ihned).

Útočník použil podvržený požadavek, ten byl ale po zpracování systémem okamžitě odstraněn. Klientský počítač je ochráněn.

V případě podvržené odpovědi je situace následující:

- (Cíl útočníka odeslal ARP dotaz na bránu/router.)
- Příchozí odpověď je zkontrolována, zda je vůbec očekávána.
- Očekávaná odpověď je povolena, neočekávaná je zamítnuta.
- Systém čeká určitý interval na případné další odpovědi na tentýž dotaz (v neurčeném pořadí v případě útoku odpoví jak oprávněný směrovač, tak i útočník).
- Systém v případě více odpovědí od různých zdrojů spouští analýzu dosavadního provozu a hledá případné útočníky.

Předpokládáme, že útočník je na síti aktivní, tedy ARP útoku předchází zjišťování prostředí, přičemž útočník používá platnou IP adresu podsítě. To způsobí, že napadený eviduje dvě různé IP adresy (svoji vlastní a navíc adresu, za kterou se vydává) pro totožnou MAC adresu. Na základě této analýzy je schopen rozlišit mezi útočníkem a skutečným nositelem fyzické adresy. Způsob detekce je detailněji popsán v implementační části práce.

Výše uvedený předpoklad ovšem obsahuje menší podmínku. Útočník totiž nemusí na lokální síti vůbec vystupovat pod aktivní IP adresou, IP adresu cíle může znát z předchozího zkoumání prostředí. Pro omezení tohoto typu útoků by bylo potřeba kontrolovat, zda dotazované stanice odpovídají. Pokud by stanice neodpověděla, byla by považována za neaktivní. V případě analýzy případného útoku by se označila za neplatnou a odstranila. Nevýhodou je, že ARP paket nemusí vůbec příjemci dorazit, protože ARP neposkytuje zaručený provoz. Další možností je uchovávat časové značky a dávat přednost starším a

známým MAC adresám před nově přidanými, ovšem je potřeba zachovat správnou funkčnost sítě při dynamickém rozdělování IP adres (DHCP[4]). Jako alternativa může fungovat i počítaní obdržených paketů, kdy útočník bude odesílat větší množství paketů, případně začne útok odesláním dotazů na všechny IP adresy podsítě. My však budeme předpokládat, že útočník chce využít nasbírané informace k okamžitému průniku do počítače oběti a tak bude operovat s funkční IP adresou na lokální síti. Pro detekci samotného útoku je navržená funkcionalita dostačující. Navíc v případech, kdy útočník operuje též s platnou IP adresou, budeme schopni jej i odstavit.

3.2 TCP

Protokol TCP je síťový protokol balíku TCP/IP[23] operující na transportní vrstvě ISO OSI modelu, nad protokolem IP(v4) nebo IPv6[9]. Jedná se jeden ze základních protokolů pro ustanovení spojení mezi dvěma internetovými uzly.

3.2.1 Použití protokolu

Protokol TCP se používá pro univerzální spojení dvou počítačů (klienta k serveru), protože na rozdíl od protokolu IP obstarává navíc:

- ustavení spojení,
- potvrzované doručení,
- pořadí paketů,
- detekci a případná eliminaci zahlcení[20] [19],
- více služeb najednou (pomocí portů).

Schopnost TCP přenést data oběma směry bez nutnosti zásahu z aplikační vrstvy stojí za jeho masivním nasazením mezi internetovými aplikacemi (web, email), přenosem souborů (FTP, P2P) aj. Technická vybavenost protokolu může způsobovat časové prodlevy v přenosu, proto nasazení pro spojení v reálném čase není příliš vhodné. Stejně tak nelze TCP použít pro broadcast či multicast (odeslání všem/více příjemcům).

Komplexnost protokolu dokládá i jeho stavový diagram[23] na obrázku 3.2, jedná se o variantu bez uvedení časových intervalů a bez ošetřených chybových stavů.

Aby došlo ke spojení dvou počítačů, je nutné, aby na přijímací straně běžel TCP server na určitém portu a na odesílací straně, která zahajuje spojení, TCP klient z určitého portu informace odesílal. Před samotným přenosem dat se ustanoví `session` (sezení), což odpovídá třem krokům:

- Odesílací strana (klient) odešle pokyn k synchronizaci.
- Příjemce (server) potvrdí přijatý požadavek a odešle svůj synchronizační příkaz.

- Klient potvrdí synchronizaci se serverem a sezení je ustanoveno.

Jakmile je server i klient v jednom sezení, dojde k přenosu dat. Ta se posílají po určitých částech, které se postupně posouvají nad odesílacím seznamem, každá část musí být druhou stranou potvrzena jako přijatá. (Velikost výsledného paketu je ale ovlivněna MTU parametrem linky a následnou fragmentací protokolu IP.) Informace vztahující se k jiné části jsou zahozeny a odesílatel je vyzván k synchronizaci. Obsah ztracený cestou, který v dané části příjemci chybí, je znovu od odesílatele vyžádán. Po dokončení přenosu dat se spojení začíná ukončovat (z klientské či ze serverové strany). Později ještě mohou docházet „zatoulané" pakety, což je důvod proč se sezení okamžitě neukončuje, ale přechází do vyčkávací fáze, než bude úplně zavřeno. Některé implementace, aby se vyhnuly této čekací fázi a držení systémových prostředků, odesílají těsně před přechodem do ní informaci o násilném ukončení spojení, čímž se spojení okamžitě zruší. To řeší problém zvýšených nároků na systémové zdroje, pokud je linka přetížená, čímž nedochází ke korektnímu ukončení sezení a OS drží prostředky těchto spojení zbytečně.

3.2.2 Obsah paketu

Obrázek č. 3.3 ukazuje paket TCP[17]. Vyobrazený paket neobsahuje data z protokolů nižších vrstev (Ethernet, IP). Pro firewall jsou důležitá téměř všechna pole paketu. Stručný popis jednotlivých polí:

- Zdrojový a cílový port: označují službu a používají se k rozlišení jednotlivých spojení.
- Sekvenční číslo: relativně označuje pozici aktuálních dat v celkovém datovém souboru.
- Potvrzovací číslo: potvrzuje přijatou část až do relativně uvedeného bytu.
- Odsazení dat: určuje začátek dat v paketu.
- ECN: explicitní hlášení zahlcení.
- Kontrolní bity: označení funkčního významu paketu (synchronizace [SYN], potvrzení [ACK], reset [RST], ukončení [FIN], uživatelská data [PSH], urgentní [URG]).
- Příjmové okno: možnosti přijetí dat na straně odesílatele.
- Kontrolní součet: kontrolní výpočet nad IP a TCP hlavičkou a daty.

Provázanost TCP s IP či IPv6 je natolik úzká, že firewall si musí uchovávat informace z obou protokolů pohromadě. Jen tak je možné udržovat tzv. endpointy, tedy jednoznačné označení pro spojení obsahující čtveřici IP (v4/v6) adresa klienta i serveru a port klienta a serveru.

3.2.3 (Ne)bezpečnost protokolu

Srovnáme-li TCP a ARP, je jasné, že díky stavům v TCP je útok obecně složitější, protože bude potřeba dodržovat některá kritéria, na druhou stranu složitější systém nám poskytuje vícero míst využitelných pro útok. Odmyslíme-li si omezení kladená stavovým konečným automatem TCP je jeho samotné zabezpečení víceméně žádné, není k dispozici žádné ověřování původu paketu.

3.2.3.1 Ukradení sezení

Pod pojmem ukradení sezení[21] máme na mysli narušení sezení útočníkem do takové míry, že jím podstrčená data jsou zpracována jako korektní část přenášené informace. K úspěšnému podvržení paketu je potřeba sledovat dosavadní spojení a sestavit TCP paket se správně vytvořenými sekvenčními a potvrzovacími čísly. Podle načasování mohou nastat dvě reakce: paket se podaří zařadit do sledované komunikace a jedna ze stanic se bude resynchronizovat, nebo obě stanice se desynchronizují, čímž se spustí útok desynchronizace obou stanic (popsán dále).

3.2.3.2 Desynchronizace klienta i serveru

Pokud se uvnitř komunikace objeví vložený paket, který způsobí, že server i klient jsou desynchronizováni, dochází k tomu, že obě strany začnou posílat resynchronizační požadavky, které se už nikdy nezesynchronizují. Dojde k zahlcení sítě do té doby, než se jeden z paketů ztratí a spojení se zruší.

3.2.3.3 Resetování spojení a synchronizace mimo úvodní fázi

Základem napadení je dosavadní sledování provozu na síti. Sestavením RST paketu s validními sekvenčními a potvrzovacími čísly (tj. ta, která padnou do aktuálního přenášeného okna) odesílatel způsobí uzavření napadeného spojení. Toto chování lze navodit i s použitím SYN paketu (který je povolen pouze při ustanovování spojení).

3.2.3.4 Zahlcení synchronizačními požadavky

Velmi jednoduchý útok[18] spočívá v polovičním navázání spojení. Útočník odešle SYN, příjemce (oběť) odpoví odesláním SYN+ACK, přičemž v systému přidělí prostředky pro toto nově vznikající spojení. Masivním zahlcením dalšími pakety SYN se docílí stavu, kdy příjmová strana již nemá systémové prostředky pro nově vznikající spojení a přestane odpovídat. Pokud útočním v paketu vyplní cizí IP adresu, nejde jej vysledovat a navíc, je-li za ní funkční stroj, pak ten na odpověď SYN+ACK odpoví RST, čímž ještě zvyšuje zátěž napadeného počítače. Obdobná varianta spočívá v navazování spojení s podvrženou IP adresou, kterou používá sama oběť, čímž dojde k vyčerpání prostředků mnohem rychleji.

3.2.3.5 Přetečení a podtečení paměti ovladače TCP

Za útokem stojí softwarová chyba (neošetřené vstupy od neověřených zdrojů), nikoliv přímo funkcionalita TCP. Využívá se pouze jeho struktury k napadení softwarové části operačního systému zodpovědné právě za TCP/IP. Technika, která může dostat útočníka až do napadeného systému, spočívá v sestavení speciálně vytvořeného paketu. Systém na straně příjemce obsahuje softwarovou chybu a zákeřný paket ji využije takovým způsobem, který systém odstaví nebo se jej (zčásti) zmocní.

3.2.4 Ochrana před útoky

Obrana před některými útoky je složitá, případně natolik výpočetně či časově náročná, že se k ní v praxi nepřistupuje. Některé útoky tak nelze řešit v rámci protokolu TCP. Na řadu se tak dostávají řešení z vyšších vrstev ISO OSI. Jedná se zejména o techniky poskytující záruku celistvosti a nenarušenosti obsahu, případně doplněné o ověřitelnou identitu odesílatele. Jmenujme např. zapouzdřování paketů IPSec[8]. Základem každé prevence by mělo být znesnadnění odposlechu spojení a vysledování položek sekvenčního a potvrzovacího čísla.

Dalším obecným prvkem prevence je korektní a striktní nastavení síťových prvků. Některé obecné přístupy:

- Zakázání odchozí komunikace z IP adres, které se nevyskytují v odesílací podsíti (před routerem).
- Zakázání příchozích paketů, které mají zdrojovou adresu patřící do cílové podsítě (za routerem).
- Bezpečná typologie sítě (oddělení veřejných služeb od privátních zdrojů) a využití filtrovacích prostředků poskytovaných síťovým hardwarem.
- Udržování aktualizovaného operačního systému a poskytovaných služeb.
- Omezení kvantitativního provozu na síti generovaného jedním prostředkem.
- Použití doporučených bezpečných konfigurací pro síťové služby a vypnutí ladících informací (či funkcí) na produkčních systémech.

Některá řešení obrany před útoky pomocí TCP porušují standardem vynucené chování (omezují povinnou funkčnost nebo zavádějí novou nestandardní funkcionalitu). Obecně by mělo platit, že pokud nestandardní chování zabezpečeného produktu způsobí nefunkčnost s nezabezpečeným standardním produktem, taková ochrana by se neměla použít. Rizikem nedodržení tohoto pravidla je existence několika zabezpečených, bohužel ale nekompatibilních, řešení. Ve výsledku tedy situace zcela kontraproduktivní.

3.2.4.1 Ukradení sezení

Jelikož TCP neumí odlišit regulérního a útočícího odesílatele, je nutné vnést do uživatelských dat přidanou informaci, která by zajistila alespoň kontrolu přeneseného obsahu.

3.2.4.2 Desynchronizace klienta i serveru

Po desynchronizaci obou stran není možné použít „ne-ACK“ prostředek pro opětovnou synchronizaci. Ale můžeme tento stav detekovat a zareagovat zrušením spojení a zabránit tak přehlcení sítě.

3.2.4.3 Resetování spojení a synchronizace mimo úvodní fázi

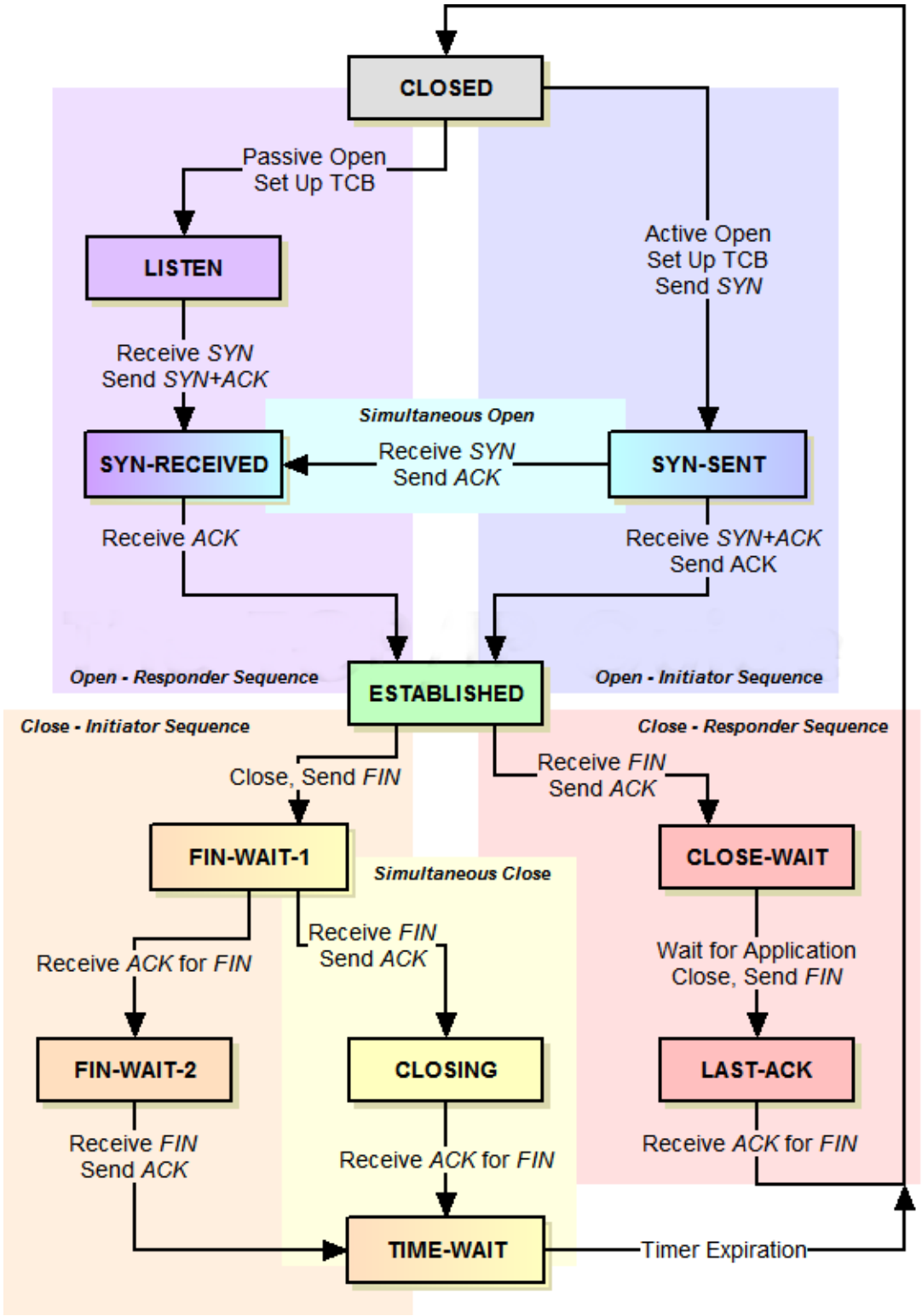
U podvrženého paketu lze ověřovat, zda je možné, aby přišel z daného rozhraní, nebo zda existuje síťová cesta k odesílateli. Zatímco SYN paket mimo část ustanovení TCP spojení lze ignorovat, u RST paketu je situace složitější. Pokud by i po RST přicházely pakety zapadající do původního spojení, je pravděpodobné, že RST paket není od odesílatele, ale od útočníka, protože v případě správného RST paketu by odesílatel nadále posílal už jen RST pakety.

3.2.4.4 Zahlcení synchronizačními požadavky

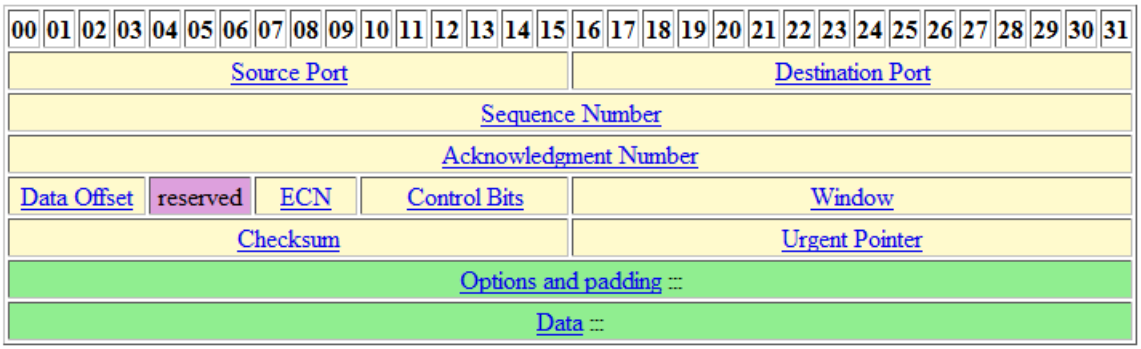
Minimalizace prostředků nutných při ustanovování spojení a menší časové limity jsou jedním z možných řešení. Ke snížení možnosti napadení může vést i použití početního (a časového) omezení při dosažení limitu počtu přijatých paketů. Proti útoku lze postavit i transparentní proxy, která by spojení ustanovovala sama. V případě navázání spojení by ho proxy replikovala původnímu cíli a tím předala původnímu adresátovi, nedokončená spojení by se odstraňovala.

3.2.4.5 Přetečení a podtečení paměti ovladače TCP

Jak již bylo zmíněno, nejedná se přímo o útok pomocí funkcionality protokolu TCP, ale o zneužití chyby v části systému, která TCP protokol spravuje. Jediným řešením je důsledná kontrola vstupních informací, zejména kontrola polí paketů označujících délku oproti reálné velikosti přijatého paketu. Účinným prostředkem proti napadení může být i dynamická kontrola výskytu podezřelých hodnot, které označují na cílovém počítači napadnutelnou část paměti. Součástí ochrany může být i kolekce vzorků závadných paketů cílených na určitá místa v systému, se kterou lze aktivně hledat podezřelé pakety pouze na základě několika znaků v jejich struktuře.



Obrázek 3.2: Stavový diagram TCP



Obrázek 3.3: Paket TCP

Kapitola 4

Implementace firewallové knihovny

4.1 Protokol ARP

4.1.1 Popis struktur a tříd

V souborech `avgfwarpendpoints.*` je hlavní implementace filtrování ARP protokolu. Součástí souboru jsou:

- definice MAC adresy (`AvgFwARPMacAddressItem`),
- položka vyrovnávací paměti ARPů (`AvgFwARPCacheItem`),
- `endpoint`, jednoznačná identifikace spojení (`AvgFwARPEndpointMapItem`),
- a hlavní třída (`AvgFwARPEndpointMapItem`).

Jednotlivé položky jsou soustředěny do map pojmenovaných `AvgFwARP...Map`. Mapy jsou navrženy jako bezpečné pro vláknové zpracování a jednotlivé objekty nesou informaci o referenci, čili jsou též vhodné pro zpracování více vlákny.

V rámci zpracovávání ARP paketů narazíme na několik časových limitů:

- `maxArpEndpointTime` je maximální doba, po kterou budeme udržovat otevřený endpoint,
- `maxArpCacheItem` je maximální doba, po kterou budeme udržovat záznam ARP cache,
- `maxMacAddressBlockTime` je (konfigurovatelná) doba, po kterou budeme blokovat útočníka podle jeho MAC adresy.

Protože se časové intervaly mezi jednotlivými verzemi operačních systému Windows NT liší, jsou použity nejmenší shodné intervaly. Aktuální hodnoty výše uvedených intervalů jsou v daném pořadí následující: 10 sekund, 15 sekund a 7 minut.

4.1.2 Hlavní funkce

Metoda `ProcessPacket` poskytuje hlavní filtrovací funkcionality. Jejími vstupy jsou:

- `action`: číslo určující výslednou akci pro daný paket,

- `arpTableOperation`: identifikační číslo, pomocí kterého se rozlišuje, jestli je nutné nějaká operace nad ARP tabulkou[1],
- `directArp`: dvoustavová hodnota, která určuje, jestli příchozí ARP paket byl doručen přímo na naši MAC adresu,
- `data`: struktura naplněná jednotlivými poli z ARP paketu (viz část 3.1.2).

Při příchodu ARP paketu do systému připraví NDIS část firewall ovladače vstupní parametry, provede kontrolu a naparsování paketu a zeptá se metody `ProcessPacket`, zda má NDIS paket povolit či zakázat.

Po předání řízení metodě `ProcessPacket` se zkontrolují následující vstupní podmínky:

- obecné kontroly odpovídající kontrole vstupů tak, jak stanoví programovací konvence,
- data paketu obsahují informace o ARP paketu,
- ARP paket operuje nad protokolem IP.

Po úvodních kontrolách se provedou operace podle směru paketu (odchozí ze systému nebo příchozí do systému) a jeho typu. Jednotlivé varianty popisují následující odstavce.

4.1.2.1 Odchozí ARP dotaz

Zkontrolujeme, zdali se v polích zdrojová a protokolová adresa (v našem případě odpovídající IP adresám) data shodují. Pokud ano, znamená to, že se jedná o `Gratuitous ARP`. Podle parametru `directArp` zjistíme, jestli se jedná o naši MAC adresu. Jsou-li splněny obě podmínky, znamená to, že naše stanice rozesílá ARP paket, ve kterém sděluje okolním stanicím svoji IP adresu. Pokud se v podsíti nachází jiná stanice, která má aktuální IP adresu shodnou s tou naší, obdržíme na náš dotaz odpověď. Následně náš operační systém dostane informaci, že používáme duplicitní IP adresu. Tu uvolní, příp. informuje server pro dynamické přidělování adres (DHCP server) o kolizní adrese.

Pokud se zdrojová a cílová adresa liší, odesíláme standardní ARP dotaz. Projdeme seznam existujících odeslaných odpovědí (označujeme je endpointy, tedy cílová místa v systému). Pokud se dotaz v seznamu nachází, zvýšíme počet očekávaných odpovědí o jeden, v opačném případě vytvoříme nový endpoint s počtem očekávaných odpovědí roven jedna. V obou případech nastavíme časovou platnost endpointu `maxArpEndpointTime`. V případě, že časový limit vyprší, endpoint je odstraněn.

1. Tato operace musí probíhat mimo prostředí jádra. Existuje možnost měnit ARP tabulku přímo v ovladači TCP/IP v systému, ale pouze s použitím nedokumentovaných funkcí, což je cesta, kterou jsme označili jako nepřípustnou.

4.1.2.2 Odchozí ARP odpověď

Odchozí odpověď není nutné nijak filtrovat (na úrovni ARP filtrování; obecné filtrování FW může stanovit odlišná pravidla, která toto rozhodnutí mohou následně změnit), protože nepředstavuje pro náš systém bezpečnostní riziko. MAC adresa příjemce se nikam neukládá.

4.1.2.3 Příchozí ARP dotaz

První kontrolou je zjištění MAC adresy odesílatele mezi blokovanými MAC adresami. V případě evidování v tomto seznamu se paket odmítne.

Příchozí dotazy povolíme, ale protože se jedná o nedůvěryhodné zdroje, ihned po zpracování dotazu odesílatele smažeme ze systémové ARP cache. Ať již se jednalo o útočníka nebo regulérní stroj, budeme spoléhat jen na ty MAC adresy, které nám ostatní sdělí pouze na naše ARP dotazy (blíže v další části).

4.1.2.4 Příchozí ARP odpověď

Jako první se zkontroluje, zda odesílatel není blokován podle jeho MAC adresy. Pokud je zablokován, aktuální paket se nepustí do systému.

Pokud paket nese informaci o `Gratuitous ARPu`, pak musíme ověřit, jestli paket dorazil přímo na naši MAC adresu. V kladném případě musíme paket pustit, protože na síti operují dva počítače se stejnou IP adresou, což musíme systém sdělit. V opačném případě musíme paket zablokovat (pochází od nedůvěryhodného zdroje). Na tomto místě můžeme odstranit existující záznamy z naší ARP cache (nikoliv systémové), ale pro možnost zneužití pro odstavování jiných strojů z ARP cache se této možnosti vyhneme. Bezpečnostní dopady jsou minimální. V případě nové stanice na síti budeme dostávat nové ARP dotazy a k aktualizaci záznamů dojde záhy.

Další kontrola v pořadí je zjištění, zda vůbec na příchozí odpověď čekáme. To se dozvíme z iterace seznamu endpointů, kde budeme hledat odpovídající ARP dotaz. Najdemeli ho, snížíme počet očekávaných odpovědí o jednu a uložíme si parametry z paketu do naší ARP cache. Pokud již existuje starší záznam v ARP cache, je označen jako neplatný, ale stále je uchováván po dobu platnosti cache záznamu, nový je přidán. Důvodem je fakt, že netušíme pořadí ARP odpovědí, zda půjde o podvržený a korektní, nebo o korektní a podvržený. Jestliže se počet očekávaných odpovědí dostal do záporných čísel, musíme zkontrolovat, zda se nám útočník nepokouší podstrčit ARP odpověď. Dostáváme se do situace, kdy na `x` dotazů jsme získali minimálně `x+1` odpovědí. Když tato kontrola není nutná, paket povolíme.

4.1.2.5 Kontrola potenciálně napadené ARP cache

Aktuální paket může nahrazovat korektní paket od jiného stroje, proto vrátíme počet očekávaných odpovědí na původní hodnotu (abychom si nezablokovali odpověď ze správného počítače, když podle počtu odeslaných dotazů nastavujeme tento parametr).

V naší ARP cache vyhledáme záznamy se stejnou MAC adresou (odesílatele) jakou má aktuální paket. Najdeme záznamy s MAC adresami útočníka, kdy jednou použil svoji vlastní IP adresu a podruhé IP adresu jiného počítače. IP adresa aktuálního paketu označuje, který z dvou výše uvedených záznamů patří útočníkovi. Útočníkem tedy je počítač, u kterého evidujeme pro jednu jeho MAC adresu několik IP adres, přičemž jedna z těchto IP adres je konfliktní v naší ARP cache.

Nyní prohledáme naši ARP cache a to včetně neplatných záznamů. Mohou nastat dvě situace:

- ARP cache nese neplatný záznam korektní dvojice MAC+IP a aktuální paket je útočníka a nese podvrženou kombinaci MAC+IP (útočník se nedostal do systému),
- ARP cache nese podvrženou neplatnou kombinaci MAC+IP a aktuální paket je korektní pár MAC+IP (útočníkův paket byl již zpracován systémem).

V případě prvním, kdy útočníkův paket právě zpracováváme, korektní neplatný záznam v cache obnovíme a aktuální útočníkův paket zablokujeme. V druhém případě podvržený neplatný záznam v cache odstraníme a aktuální korektní paket povolíme.

Nakonec přidáme útočníkovu MAC adresu do seznamu zablokovaných adres, čímž znemožníme případné další pokusy o hlídání napadené cache.

4.1.3 Omezení v implementaci

Jak již bylo zmíněno v části 3.1.4, stávající implementace neposkytuje ochranu v případě, že útočník nepracuje na síti též se svoji vlastní IP adresou. Takto jednající útočník nebude nalezen v seznamu MAC+IP párů, kde MAC adresa patří útočníkovi a IP adresy se liší (napadená a originální). Avšak i v tomto případě jsme schopni útok alespoň ohlásit uživateli, který má možnost incident nahlásit správci sítě a omezit citlivou komunikaci, používat šifrování či přepnout firewall do striktnějšího režimu.

Další nevýhodou je pořadí paketů od útočníka a původního počítače. Může se totiž na velmi krátkou chvíli stát, že útočníkova MAC adresa je použita jako korektní, a to až do příchodu korektní ARP odpovědi. A právě útočník bývá blíže napadenému místu a může odpovídat rychleji.

Pro zamezení ARP útoku je nutné, aby výše popsaný systém prevence používala jak potenciální oběť, tak i počítač, se kterým navazuje komunikaci, která je právě předmětem zájmu útočníka. Pokud tento druhý počítač není schopen ARP útoku zamezit, útočník i přes ochranu na straně oběti je schopen odposlouchávat provoz ve směru od napadeného počítače k oběti.

Některé speciální aplikace nebo speciální konfigurace sítě mohou způsobit, že systém ochrany před ARP útokem chybně vyhodnotí korektní počítač jako hrozbu. To se stane za situace, kdy pro dvě MAC adresy na síti bude existovat pouze jedna IP adresa. Počítač s nainstalovanou ochranou odešle ARP dotaz na tuto IP adresu a zpátky mu odpoví oba nositelé této IP adresy, každý se svojí MAC.

4.2 Protokol TCP

4.2.1 Popis struktur a tříd

Soubory `avgfwendpoint.*` a `avgfweptable.*` implementují filtrování TCP protokolu. Mezi definicemi najdeme:

- definici TCP endpointu, jednoznačného označení TCP spojení (`AvgFwEndpoint`),
- třídu nesoucí všechny TCP endpointy (`AvgFwEPTable`).

Jednotlivé endpointy se opět ukládají do map pojmenovaných `AvgFwEndpointMap`. Mapy jsou vícevláknově bezpečné a jednotlivé endpointy nesou informaci o počtu referencí, čili se taktéž dají použít ve vícevláknovém zpracování.

Hlavní třída `AvgFwEPTable` nese tři mapy endpointů:

- mapu pro regulérní endpointy[2] (informace o aktivním spojení, detailněji popsán níže),
- mapu pro poslouchající servery[3] (informace o přípustných spojeních dovnitř počítače),
- mapu pro endpointy ve vyčkávací fázi[4] (informace o ukončovaných spojeních).

`AvgFwEPTable` je zapojena do systému TDI a NDIS pomocí několika metod. Pro filtrování paketů TCP (z NDIS části), se používá metoda `ProcessPacket`. Pro obsluhu TCP událostí z TDI se používají metody:

- `CreateRegularEndpoint` pro událost `TDI_CONNECT` (připojení ven) a `TDI_ACCEPT` (připojení dovnitř),
- `CreateListenEndpoint` pro událost `TDI_LISTEN` (poslouchání na portu),
- `DeleteListenEndpoint` pro událost `TDI_DISASSOCIATE_ADDRESS` (ukončení poslouchání na portu),
- `DeleteRegularEndpoint` pro událost `TDI_DISCONNECT` (ukončení spojení).

Specifikace TCP zmiňuje ve zpracování TCP paketů časové limity. Tyto limity se mezi jednotlivými výrobci TCP ovladačů liší, a proto v implementaci jsou použity nejčastější nebo nejbezpečnější varianty:

- `maxIdleEndpointTimeout` je maximální doba, po kterou budeme udržovat endpoint ve fázi před ustanovením sezení,

2. Endpointy v TCP stavech: SYN-SENT, SYN-RCVD, ESTABLISHED, CLOSE-WAIT, LAST-ACK, FIN-WAIT1, FIN-WAIT2, CLOSING.
3. Endpointy v TCP stavu LISTEN. V implementaci došlo ke zjednodušení sloučením se stavem CLOSED, byl vytvořen jeden stav CLOSED-LISTEN, o stavu endpointu tak rozhoduje jeho umístění v daném seznamu.
4. Endpointy v TCP stavu TIME-WAIT.

- `maxUnfilteredEndpointTime` je maximální doba, po kterou budeme držet endpoint ve vyčkávací fázi,
- `maxFinIdleEndpointTimeout` určuje maximální čas, po který budeme držet endpoint ve fázi ukončování spojení.

TCP endpoint, jako jednoznačný identifikátor spojení, je tvořen následujícími položkami:

- IP adresa a port zdrojového počítače,
- IP adresa a port cílového počítače,
- číselná identifikace aplikace (ozn. `PID`; je-li dostupný z TDI části),
- prvotní spojení v sezení a jeho směr,
- aktuální TCP stav podle stavového diagramu 3.2,
- výsledná akce pro filtrování.

Součástí zdrojového kódu je i podpora pro jednoduché routování pomocí služby systému Windows nazývané `Sdílení připojení k Internetu`. Pokud je zapnuté, odchozí pakety připojených klientů se objevují bez předchozího průchodu vrstvou TDI a tedy nejsou svázány s konkrétní aplikací. Pro udržení co nejvíce restriktivního pravidla pro systémovou komunikaci (na úrovni paketů) je filtrační mechanizmus schopen detekovat a správně nastavit pravidla pro rozsahy podsítí vymezených pro Sdílení připojení.

Firewall podporuje zobrazování dotazů uživateli v případě výskytu neznámé komunikace, pro kterou neexistuje pravidlo v konfiguraci. Z tohoto důvodu je nutné některé pakety zdržet, dokud uživatel nerozhodne, jak s nimi má firewall naložit (povolit či zakázat). Toto zdržení je implementováno tak, že dotčené pakety se uchovají do doby rozhodnutí pod TCP endpoint. Maximální možná doba takového uchování je definována systémem Windows, v kódu je uvedena jako `maxPendedEndpointTime`.

4.2.2 Zpracování TCP paketu

Metoda pro zpracování paketu (`ProcessPacket`) má následující vstupní parametry:

- `action`: výsledná akce pro daný paket (povolit/zakázat),
- `pid`: číselné označení aplikace, které paket patří,
- aktuální pravidla a rozsahy sítí,
- rozšířené informace ze zpracování (např. použij akci `A`, nová systémová komunikace, režim routování aj.),
- `request`: TCP parametry spojení (adresy, porty apod.),

- data: kopie dat z polí TCP paketu (viz obrázek 3.3).

Jakmile do systému dorazí nebo ze systému se odešle TCP paket, NDIS část firewall ovladače vytvoří vstupní parametry, provede kontrolu a naparsování paketu a zeptá se metody ProcessPacket, zda má NDIS paket povolit či zakázat.

Po zavolání metody `ProcessPacket` se paket pokusíme spojit s některým z endpointů (resp. s jejich informacemi o prvotním TCP požadavku) v seznamech v tomto pořadí:

- regulérní endpointy,
- poslouchající endpointy,
- poslouchající endpointy (použije se lokální IP adresa `0.0.0.0`),
- vyčkávající endpointy.

Tyto seznamy se v pravidelných intervalech prochází a na základě stavu podle TCP stavového diagramu se použije určitý časový limit pro kontrolu, zdali spojení již nemá být ukončeno. V případě vypršení limitu je endpoint odstraněn.

4.2.2.1 Zpracování přes regulérní endpoint

Filtrování odchozího TCP zahrnuje dvě varianty. První počítá s tím, že odchozí TCP jde přes TDI vrstvu, provoz pochází od aplikace. Druhá varianta vzniká při komunikaci přímo z operačního systému. Tyto přístupy se liší tím, že pro TDI máme informaci o TCP provozu ještě před zachycením na NDIS vrstvě, zatímco u druhé možnosti stojí TDI mimo, pracuje pouze NDIS část.

Odchozí TCP přes TDI a NDIS vrstvu Při pokusu aplikace o odeslání dat pomocí TCP se na TDI vrstvě vyvolá událost `TDI_CONNECT`. Událost je zachycena knihovnou firewallu a začíná její zpracování. Nejprve je nutné ověřit, že v systému není stejný požadavek veden u vyčkávajících endpointů. Pokud by byl, bude odstraněn. Poté se vytvoří nový endpoint (`CreateRegularEndpoint`), kterému se nastaví hodnota pro identifikaci aplikace (`PID`). Událost se vrátí ke zpracování TDI s výsledkem „povolit“.

Data následně zpracuje systém a vytvoří odpovídající pakety. Ty dorazí k ověření do NDIS části firewall knihovny do hlavní metody `ProcessPacket`. Zkontrolují se vstupní podmínky:

- obecné kontroly odpovídající kontrole vstupů tak, jak stanoví programovací konvence,
- data paketu obsahují informace o TCP paketu.

Odpovídající endpoint se najde v seznamu regulérních endpointů, vyresetuje se jeho časová značka, spustí se kontrola polí TCP paketu oproti stavovému diagramu TCP. Pokud se aktuální stav TCP změnil na `TIME-WAIT` nebo `CLOSED`, endpoint se přesune mezi vyčkávající

endpointy. Pokud je s endpointem svázaná nějaká aplikace (`PID` není nulový), uloží se tato informace do struktury nesoucí data spojení, protože na NDIS vrstvě nemáme vlastníka paketu rozlišeného (resp. vlastníkem paketu je vždy systém). Nakonec se použije uložená akce z endpointu jako výsledek celé operace[5].

Odchozí TCP pouze přes NDIS vrstvu V některých případech se může stát, že odchozí TCP spojení mine vrstvu TDI, resp. volání pochází z nižších vrstev systému (komunikuje přímo operační systém nebo nějaký ovladač, např. VPN). Pro firewall je tato situace mírně problematická, protože ačkoliv provoz může být řízen nějakou aplikací, zdrojem dat je součást systému, a proto nelze k paketům korektně přiřadit aplikaci jako vlastníka. To je důvod, proč se tato komunikace bude vždy označovat jako systémová.

Odchozí paket je přímo odeslán firewallové knihovně k ověření pomocí `ProcessPacket` metody. Endpoint pro tuto komunikaci zatím neexistuje, takže paket zkontroluje, zdali nese příznak SYN a případně se dočasně povolí. Jakmile se projdou aplikační a systémová pravidla, máme k dispozici konečnou akci pro paket. Pokud je výsledkem „povolit“, dodatečně se ověří platnost polí TCP paketu oproti stavovému diagramu TCP a za podmínky bezchybného stavu se vytvoří nový endpoint. Další zpracování téhož provozu se již provede podle předchozí sekce 4.2.2.1.

Příchozí TCP v ustaveném sezení Příchozí pakety jsou opět předávány `ProcessPacket` metodě ke kontrole. Jelikož předpokládáme již ustavené sezení TCP, existuje k tomuto spojení odpovídající endpoint. Po kontrole hlaviček TCP paketu proti stavovému diagramu se pro paket (pokud nedošlo ke změně sítí či pravidel) použije již dříve uložená akce. Pokud došlo k posunu TCP stavu do `TIME-WAIT` nebo `CLOSED`, endpoint se přesune do seznamu s vyčkávajícími endpointy, kde podle časového limitu dojde k jejich odstranění.

4.2.2.2 Zpracování přes poslouchající endpoint

Pokud na stanici dojde ke spuštění TCP serveru na některém volném portu, firewall knihovna dostane TDI událost `TDI_LISTEN` zavoláním metody `CreateListenEndpoint`. Obsahem metody je vytvoření endpointu vyplněného pouze informacemi o identifikaci aplikace (`PID`), lokální IP adresy a lokálního portu. Takto vytvořený endpoint se zařadí do seznamu s poslouchajícími endpointy.

Jakmile dorazí příchozí TCP paket žádající spojení s tímto serverem, firewall dostane řízení do metody `ProcessPacket`, která najde k paketu s příznakem SYN odpovídající poslouchající endpoint. Poté se ověří hlavičky TCP paketu oproti stavovému diagramu a vytvoří se regulérní endpoint, který se zařadí do odpovídajícího seznamu.

Pokud TCP server ukončí svoji činnost, TDI zahlásí tento stav firewallové knihovně pomocí metody `DeleteListenEndpoint`, která poslouchající endpoint odstraní.

5. Ačkoliv to není součástí této práce, je vhodné zmínit, že součástí volání je i ověření podle pravidel a sítí, případě vznesení dotazu o povolení komunikace uživateli.

4.2.2.3 Zpracování přes vyčkávající endpoint

Pro spojení, která jsou ve vyčkávací fázi, tj. v TCP stavu `TIME-WAIT`. Firewall nalezne endpoint v odpovídajícím seznamu. Po kontrole TCP stavu a časového limitu se použije akce uložená v endpointu a paket se povolí nebo zakáže. Výjimkou je situace, kdy časový limit pro stav`TIME-WAIT` vypršel a endpoint má být odstraněn, aktuální paket může být novým spojením, bude proto v případě SYN příznaku dále zpracován.

Ve vyčkávající fázi již nedochází k přenosu uživatelských dat, proto firewall šetří systémové prostředky a nepřepočítává výslednou akci, ale používá poslední platnou uloženou v endpointu.

4.2.2.4 Kontrola dle stavového diagramu

Zjednodušená varianta diagramu, popsaná na obrázku č. 3.2, musí být rozšířena o časové limity, násilná ukončování spojení a znovu-odeslání (angl. `retransmission`) ztracených či nepotvrzených (jsou-li potvrzovány) paketů. Konkrétní implementaci nalezneme v souborech `tcpcheck.*`.

TCP stav je součástí každého aktivního endpointu (vyjma poslouchajících endpointů). Samotná kontrola začíná ověřením dostupnosti paketových informací. Následuje načtení polí příznaků, sekvenčního čísla, potvrzovacího čísla a velikosti okna. K úplné kontrole je potřeba ještě znát velikost paketu a směr paketu a mít uchovaný aktuální stav vč. případných předchozích sekvenčních, potvrzovacích čísel a délky paketu.

Z hlediska implementace bylo vhodné sloučit stavy `CLOSED` a `LISTEN`. Pro tyto stavy je totiž rozhodující umístění v seznamech endpointů, bylo tak možné vytvořit univerzální počáteční stav `CLOSED-LISTEN`.

Jako první se ověřuje přítomnost RST příznaku. Pokud podle standardu má dojít ke zpracování RST paketu, spojení se ukončuje a aktuálním stavem se stává `CLOSED-LISTEN`. V opačném případě se paket zablokuje.

Vyhodnocení v jednotlivých stavech metodami `IsAllowedIn...` je velice obdobné:

- kontrola sekvenčního čísla a potvrzovacího čísla (`TestSeqNum` a `TestAckNum`) a příznaků,
- nastavení povolených hodnot pro tato čísla v dalším paketu (`SetupByOutgoingACK`, `SetupByIncomingACK`).

Obecně řečeno, pokud standard dovoluje přechod stavu nebo znovu-odeslání paketu, návratovou hodnotou je „povolit“ a případná změna stavu se provede. V ostatních případech se pakety zakazují.

4.2.3 Omezení v implementaci

Kritickou částí implementace je přesné kopírování stavů TCP endpointů s těmi v operačním systému. Chyba v implementaci TCP endpointů nebo neohlášené změny TCP ovladače mezi

verzemi systémů mohou vést k desynchronizaci stavů a tím pádem k blokování provozu, který je z pohledu operačního systému v pořádku. Předejít tomuto lze použitím mírnějšího filtrování TCP paketů, kde budeme povinně ověřovat jenom SYN a RST příznaky a přechody mezi stavy.

Použití časových limitů ovlivňuje ve značné míře chování TCP stavů a paketů. Bohužel přesná čísla pro jednotlivé mutace systémů Windows NT nejsou k dispozici a navíc bylo měřením zjištěno, že tyto intervaly se mezi nimi i liší. Protože evidence seznamu časů je prakticky neudržovatelná, implementace počítá s univerzálními časy, což může v některých situacích omezovat jinak korektní provoz.

Rozšíření o jednu ochrannou vrstvu je přínosem pro bezpečnost systému. Na druhou stranu je další část potenciálním nositelem dalších chyb, a i přes maximální úsilí v prevenci a eliminaci chyb se může do implementace nějaká zanést, což může naopak útočníkovi práci ulehčit. Pro návrh bezpečnostního systému tak dvojnásob platí používat obecně platná pravidla pro psaní bezpečného zdrojového kódu (zejména ověřování nedůvěryhodných vstupů, ošetření všech chybových stavů a vyhrazený přístup ke sdíleným prostředkům).

Kapitola 5

Možnosti rozšíření

5.1 Ochrana před ARP útoky

V části 3.1.4 jsme popisovali omezení pro detekci a zastavení útočníka, pokud ne síti nevystupuje pod napadenou i svojí IP adresou. Pro zablokování útočníka bychom mohli použít systém známých MAC adres.

Firewall by po dobu běhu (případně z minulého spuštění) získával seznam MAC adres, které se na síti běžně vyskytují. Pokud by nenastával konflikt, do tohoto seznamu by se přidávaly i MAC adresy nových počítačů. V případě detekce útoku by firewall porovnal zainteresované MAC a IP adresy a na základě jejich dosavadního používání by stanovil, že nejméně známá MAC a IP adresa bude prohlášena za útočníka.

Zajímavým rozšířením by mohlo být použití tohoto naučeného seznamu MAC a IP adres pro vygenerování statických záznamů v ARP cache. Řešení je o trošku složitější v tom, že starší verze Windows jsou schopny přepisovat i statické položky[11].

Analýzou počtu paketů a poměru mezi ARP dotazy a odpověď'mi by bylo možné usuzovat na probíhající útok, avšak např. použitá testovací aplikace Cain & Abel[3] umožňuje nastavit intervaly odesílání podvržených paketů a tím ztížit odhalování útočníka.

5.2 Ochrana před TCP útoky

Díky technice evidování endpointů je možné velmi snadno rozšířit firewallovou knihovnu o regulaci počtu spojení. Evidováním počtu požadavků u endpointů a jejich kontrolou při jejich procházení lze při dosažení určitého limitu zamezit provozu přes tento endpoint. Problémem může být nalezení vhodných univerzálních limitů tak, aby nastavení mělo účinek i v malých sítích a přitom nedocházelo k nechtěnému blokování na větších sítích. Nevýhodou řešení je možné použití podvržené IP adresy k vynucení zablokování a tudíž zamezení přístupu k důležitému počítači na síti.

Zavedením podobného seznamu jako endpointů TCP je možné rozšířit firewall o ochranu před skenováním portů, kdy útočník se snaží najít otevřený port na cílovém počítači, identifikovat běžící službu nejlépe i s verzí spuštěného serveru a ten následně napadnout pomocí chyby přetečení paměti. Pokud klíčem pro vyhledávání bude IP adresa vzdálené strany a budeme evidovat počet zablokovaných pokusů o připojení k zavřenému portu, můžeme na určité hodnotě odesílatele paketů zablokovat podle IP adresy. Nevýhodou je opět možnost zablokování korektního počítače za pomoci podvržených IP adres.

Kapitola 6

Podobné produkty a jejich techniky

6.1 Ochrana před ARP útoky

6.1.1 ArpON

Software ArpON[2] (`ARP handler inspectiON`) je síťová služba chránící systém proti napadení ARP cache. Podporuje dva režimy provozu: statickou a dynamickou ochranu.

Stěžejní částí statické ochrany je načtení seznamu pevně konfigurovaných MAC adres. ARP dotazy a odpovědi nesoucí MAC adresu, která není v seznamu obsažena, jsou zablokovány. Po dobu běhu služby se též obnovují záznamy ARP cache v desetiminutových intervalech. Chování statické ochrany názorně přibližuje obrázek 6.1.

Dynamická ochrana pracuje na principu otevřených oken pro příjem ARP odpovědí. S odeslaným ARP dotazem se vytvoří časové okno, ve kterém je systém připraven přijmout odpověď. ARP odpovědi bez otevřeného okna jsou zablokovány. Přijaté ARP dotazy jsou zodpovězeny, ale pro svoji nedůvěryhodnost jsou z cache odstraněny jejich MAC adresy. Systém poté odešle ARP dotaz na doplnění ARP cache informací o dotazujícím se stroji. Obrázek 6.2 popisuje postup zpracování ARP přes dynamickou ochranu.

ArpON používá obdobnou techniku pro ochranu před ARP útokem, ale zapojuje i postupy, které nejsou vždy stoprocentní, např. kontrola adresy odesílatele původního ARP dotazu. ArpON k této kontrole nepoužívá takové technické prostředky, aby se zabránilo situaci, kdy korektnímu stroji paket nedorazí nebo jeho odpověď se ztratí cestou.

Fakt, že se nepoužívá ovladač pro filtrování paketů, znamená, že každý paket je systémem zpracován. V případě odstavení aplikace zůstává počítač nezabezpečen.

6.1.2 XARP

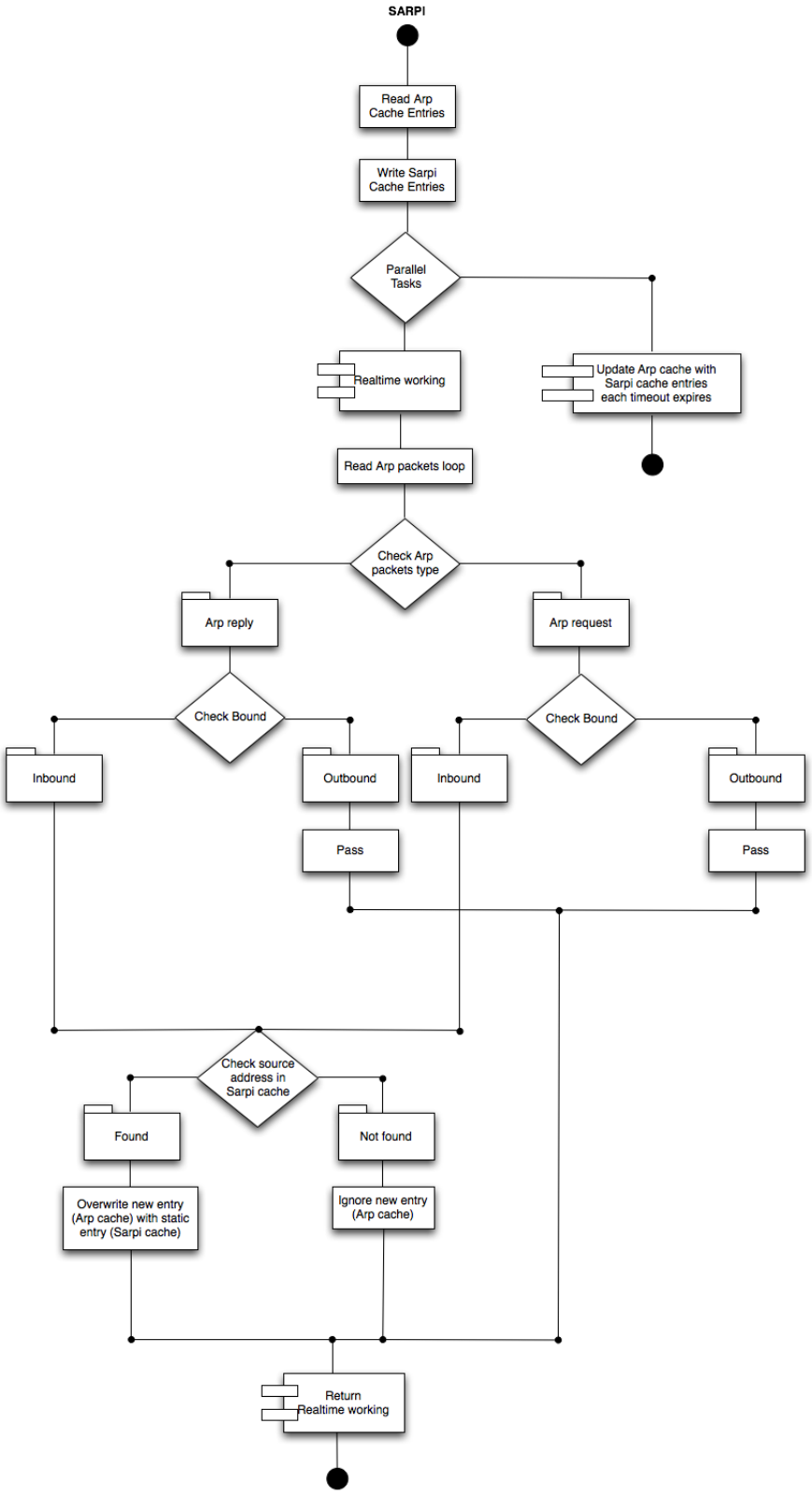
XARP[28] je nástroj sledující provoz na síti. Sbíráním informací o ARP paketech a následným porovnáním s databází validních údajů je schopen informovat uživatele o použití neznámých MAC adres.

Informace o tom, jak se program chová k novým MAC adresám, případně jak se postaví k útoku z MAC adresy považované za důvěryhodnou, autor programu nikterak nepopisuje.

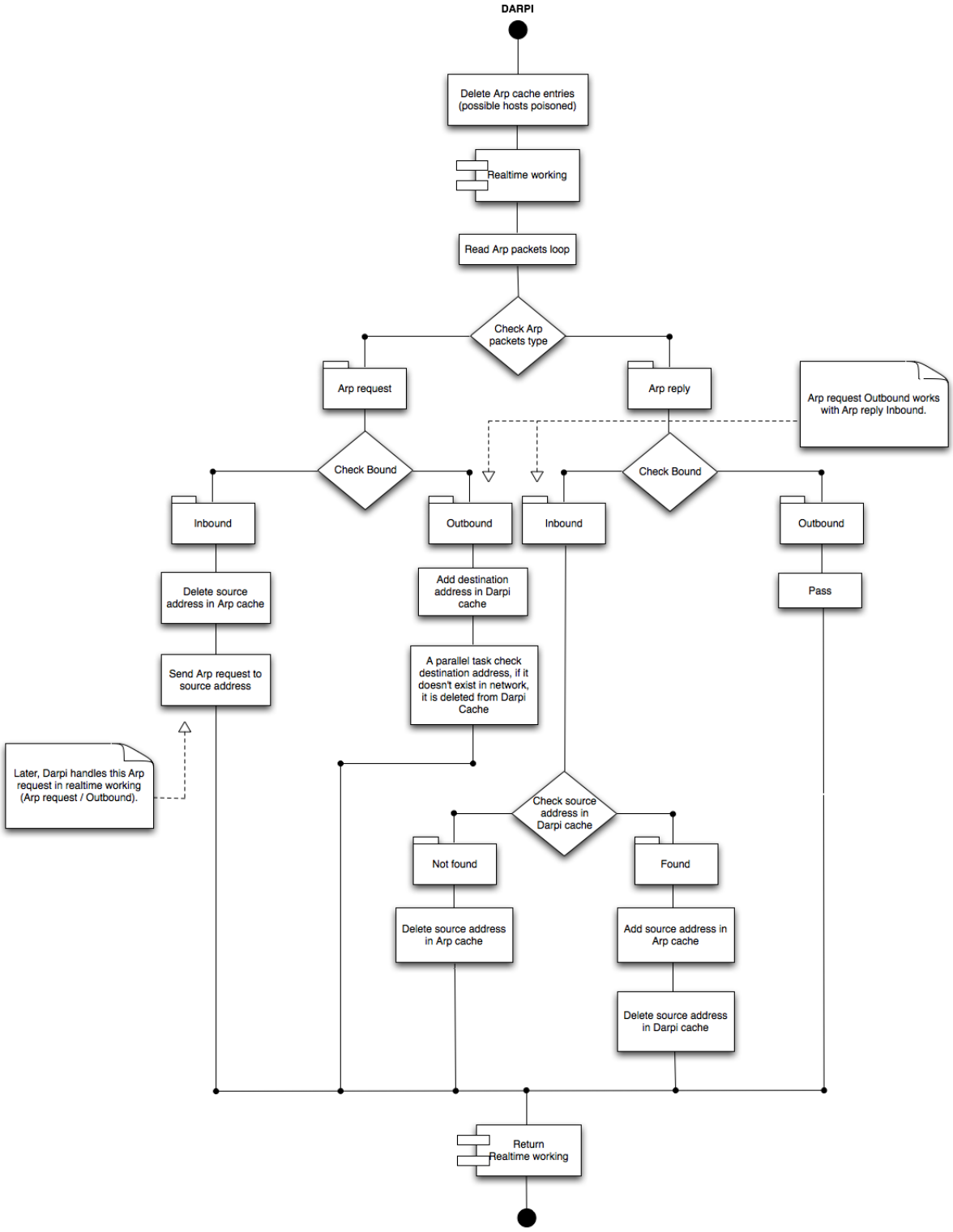
6.2 Ochrana před TCP útoky

Některým TCP útokům umí do jisté míry odolávat i systém Windows NT sám o sobě[24]. Nastavení takové ochrany se provádí v registrech pod klíčem TcpIp služby (tj. TCP/IP ovladače).

Nastavení hodnoty `SynAttackProtect` ovlivňuje časové intervaly pro znovu-odeslání SYN+ACK paketů při navazování TCP spojení. Úpravou hodnot `TcpMaxPortsExhausted`, `TcpMaxHalfOpen`, `TcpMaxHalfOpenRetried` lze nastavovat početní omezení pro polootevřená spojení (spojení ve stavu `SYN-RCVD`).



Obrázek 6.1: ArpOn - statická ochrana



Obrázek 6.2: ArpOn - dynamická ochrana

Kapitola 7

Závěr

Cílem práce bylo navrhnout a demonstrovat subsystém pro detekci a ochranu před podvrženými či neplatnými pakety protokolů ARP a TCP. V úvodních kapitolách je popsáno prostředí systému, pro které se řešení vyvíjelo.

Další kapitoly popisují protokoly ARP a TCP a jejich slabá místa, jsou jmenována některá obecná řešení. Konkrétní řešení jsou rozebrána v kapitole o implementaci. Snahou autora bylo najít taková kompromisní řešení, která neomezují funkčnost takovým způsobem, že nezabezpečená zařízení nejsou schopná komunikovat se zařízeními zabezpečenými.

Přehled podobných produktů, jejich řešení útoků a nastíněné možností rozšíření vlastního řešení ukazují, že do budoucna lze produkt dále rozvíjet a zlepšovat. Nedílnou součástí řešení jsou zdrojové kódy popisované v kapitole o implementaci a elektronické materiály demonstrující firewallovou knihovnu za běhu (viz přílohy).

Literatura

[1] Plummer, D.: *An Ethernet Address Resolution Protocol*, listopad 1982, <http://tools.ietf.org/html/rfc826> . 3.1

[2] Pasquale, A.: *ArpON*, říjen 2008, <http://arpon.sourceforge.net/> . 6.1.1

[3] Montoro, M.: *Cain & Abel*, říjen 2009, <http://www.oxid.it/cain.html> . 5.1

[4] Droms, R.: *Dynamic Host Configuration Protocol*, březen 1997, <http://tools.ietf.org/html/rfc2131> . 3.1.4

[5] Mockapetris, P.: *Domain names - implementation and specification*, listopad 1987, <http://tools.ietf.org/html/rfc1035> . 3.1.1

[6] IEEE Standard for Local and Metropolitan Area Networks, únor 2002, <http://www.ieee802.org/> . 3.1, 3.1.1

[7] Internet Protocol, září 1981, <http://tools.ietf.org/html/rfc791> . 2.2.1, 3.1

[8] Kent, S. a Seo, K.: *Security Architecture for the Internet Protocol*, prosinec 2005, <http://tools.ietf.org/html/rfc4301> . 3.2.4

[9] Deering, S. a Hinden, R.: *Internet Protocol, Version 6 (IPv6) Specification*, prosinec 1998, <http://tools.ietf.org/html/rfc2460> . 3.2

[10] ISO OSI Standard, 1994, http://standards.iso.org (ISO_IEC_7498-1_1994(E).zip) <http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip> . 2.2

[11] KB842169, červen 2004, <http://support.microsoft.com/kb/842169> . 5.1

[12] Hua, W. a Ohlund, J. a Butterklee, B.: *Unraveling the Mysteries of Writing a Winsock 2 Layered Service Provider*, květen 1991, <http://www.microsoft.com/msj/0599/LayeredService/LayeredService.aspx> . 2.2.1

[13] Microsoft Developer Network, <http://msdn.microsoft.com> . 2.2.7

[14] Divine, T.: *NDIS Developer's Reference*, leden 2010, <http://www.ndis.com> . 2.2.5, 2.2.7

[15] Finlayson, R. a Mann, T. a Mogul, J. a Theimer, M.: *A Reverse Address Resolution Protocol*, červen 1984, <http://tools.ietf.org/html/rfc903> . 3.1.1

[16] Cheshire, S.: *IPv4 Address Conflict Detection*, červenec 2008, <http://tools.ietf.org/html/rfc5227> .

[17] RFC Sourcebook (10.4), prosinec 2008, <http://www.networksorcery.com/enp/> . 3.1.2, 3.2.2

[18] CERT® Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks, září 1996, <http://www.cert.org/advisories/CA-1996-21.html>. 3.2.3.4

[19] Allman, M. a Paxson, V. a Stevens, W.: *TCP Congestion Control*, duben 1999, <http://tools.ietf.org/html/rfc2581>. 3.2.1

[20] Ramakrishnan, K. a Floyd, S. a Black, D.: *The Addition of Explicit Congestion Notification (ECN) to IP*, září 2001, <http://tools.ietf.org/html/rfc3168>. 3.2.1

[21] Bellovin, S.: *Security Problems in the TCP/IP Protocol Suite*, duben 1989, http://staff.washington.edu/dittrich/ (tcpip_problems_bellovin.ps) <http://staff.washington.edu/dittrich/talks/qsm-sec/tcpip_problems_bellovin.ps>. 3.2.3.1

[22] Davies, J. a Tausing, A.: *TCP/IP Fundamentals for Microsoft Windows*, únor 2004, <http://technet.microsoft.com/en-us/library/bb726983.aspx>. 2.2.1

[23] Kozierok, C.: *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*, 978-159327-047-6, říjen 2005, <http://www.tcpipguide.com/>. 2.2.1, 3.2, 3.2.1

[24] Meier, J. a Mackman, A. a Dunner, M. a Vasireddy, S. a Escamilla, R. a Murukan, A.: *How To: Harden the TCP/IP Stack*, leden 2006, <http://msdn.microsoft.com/en-us/library/ff648853.aspx>. 6.2

[25] Transport Driver Interface, březen 2010, <http://msdn.microsoft.com/en-us/library/ff565685.aspx>. 2.2.3

[26] Prabhala, M. a Mendoza, J.: *Windows Filtering Platform Enhancements in Windows 7*, 2008, http://download.microsoft.com (NET-T722_DDC08.pptx) <http://download.microsoft.com/download/D/1/D/D1DD7745-426B-4CC3-A269-ABBBE427C0EF/NET-T722_DDC08.pptx>. 2.2.3

[27] Windows Filtering Platform, květen 2008, <http://www.microsoft.com/whdc/device/network/WFP.mspx>. 2.2.6, 2.2.7

[28] Mayer, C.: *XARP*, březen 2008, <http://www.chrismc.de/development/xarp/>. 6.1.2

Příloha A

Přílohy

A.1 Licenční podmínky

Zdrojové kódy i binární data, pokud zákon nestanoví jinak, je možné použít pouze ke studijním nekomerčním účelům. Jiné užití je podmíněno písemným souhlasem autora.

A.2 Elektronická příloha

Nedílnou součástí této práce jsou zdrojové kódy popisované v kapitole o implementaci a demonstrační audiovizuální materiál v elektronické formě prezentující funkčnost produktu.